UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/008,952	12/06/2001	Ashley K. Wise	RA5417 (USYS030.PA)	3730

27516 7590 06/16/2005

UNISYS CORPORATION
MS 4773
PO BOX 64942
ST. PAUL, MN 55164-0942

EXAMINER	
MITCHELL, JASON D	
ART UNIT	**PAPER NUMBER**
2193	

DATE MAILED: 06/16/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

S

Office Action Summary	Application No. 10/008,952	Applicant(s) WISE, ASHLEY K.	
	Examiner Jason Mitchell	Art Unit 2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *28 March 2005*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-4 and 6-21* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-4 and 6-21* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to an application filed on 12/06/2001.

2. At Applicant's request, claims 1, 6-8, 13, and 18 have been amended, claim 5 has been canceled and claim 21 has been added. Claims 1-4 and 6-21 are now pending in this case.

Response to Arguments

3. **Except where noted, Applicant's arguments with respect to claims 1-4 and 6-20 have been considered but are moot in view of the new ground(s) of rejection.**

Regarding Claim 10: In the second full paragraph on pg. 10 of the response applicant states:

> Specifically, the teaching that COMMON LISP is designed to hide the distinction between Bignums and Fixnums suggests arithmetic operations are seamless. There is no necessary implication that the memory allocation and deallocation operators are overloaded. White does not necessarily overload language-provided memory allocation and deallocation operators, and the Office Action does not provide evidence to support this allegation

4. **Applicant's arguments have been fully considered but they are not persuasive.** Examiner's position is further supported by section 12.4 of the COMMON LISP reference 'each works on all types of numbers'. The term overloaded is generally recognized in the art to mean a function that has multiple parameter signatures. If the '+' function in lisp can take a 'fixnum' or a 'bignum', it is clearly overloaded.

Regarding Claim 12: In the paragraph bridging pgs. 8 and 9 Applicant states:

> For example, claim 12 includes limitations of identifying an optimal set of most significant bits of the dividend and a set of least-significant bits of the dividend as a function of a number of bits that represent the dividend and a number of bits that represent the devisor. The Office Action cites White's teachings at p. 178, col. 1, para. 3 in alleging that the limitations of claim 12 are anticipated. However, there is no apparent suggestion in these teachings of the specific limitations of how a set of most significant bits of the dividend and a set of least significant bits of the dividend are identified. Furthermore, there is no apparent suggestion that the identification is made as a function of the number of bits that represent the dividend and the number of bits that represent the divisor.

5. **Applicant's arguments have been fully considered but they are not persuasive.** Further support of Whites anticipation of the recited limitation can be found in the Brooks reference, which teaches the preferred algorithm used for division in the White reference (pg. 181, section 2.5).

It can be seen in step D2 on pg. 258 that the portion of the dividend used ('set of most significant bits') begins with the leftmost n digits, where n is the length of the divisor. Further the portion of the dividend not used ('set of least significant bits') is the right most m digits m being the difference between the length of the divisor and dividend (see pg. 257 '$u=(u_1 \ldots u_{m+n})$'). Therefore White discloses identifying an optimal set of most significant bits ('u_j-u_n') of the dividend and a set of least-significant bits of the dividend ('$u_n \ldots u_{m+n}$') as a function of a number of bits that represent the dividend and a number of bits that represent the devisor ('u_{m+n}').

Claim 13: In the paragraph bridging pgs. 11 and 12 Applicant states:

> Claim 13 includes limitations of identifying an optimal set of most-significant bits of the dividend and a set of least significant bits of the dividend as a function one-half a difference between the number of bits that represent the dividend and the number of bits that that represent the divisor. The cited teachings of Burnikel contain no apparent suggestion of using any difference of values to identify an optimal set of

> most-significant and least-significant bits of the dividend. Furthermore, the cited teachings suggest splitting the dividend into four parts, each part having a length of n/2.

6. **Applicant's arguments have been fully considered but they are not persuasive.** Burnikel teaches the entire dividend has a length of 2n and the divisor has a length of n, thus the 'difference between the number of bits that represent the dividend and the number of bits that that represent the divisor' is n. As Applicant has noted, each section of the dividend, including the most significant (A_1) and least significant (A_4), have a length of n/2 or half the difference. Therefore the Burnikel reference teaches identifying an optimal set of most-significant bits of the dividend (A_1) and a set of least significant bits of the dividend (A_4) as a function one-half a difference between the number of bits that represent the dividend and the number of bits that that represent the divisor (n/2).

Claim 15: In the third full paragraph on pg. 12 of the response Applicant states:

> Claim 15 depends from claim 1 and includes limitations of transferring data associated with temporary variables of the large-integer data type by moving pointers to the data. The cited portion of Anderson suggests a linked list for storage of an array, and teaches away from the claimed transferring by moving pointers. Specifically, Anderson teaches "copying of an APN to new memory area is more complex for linked structures." Thus, Anderson teaches away from transferring data by moving pointers to the data.

7. **Applicant's arguments have been fully considered but they are not persuasive.** "A known or obvious composition does not become patentable simply because it has been described as somewhat inferior to some other product for the same use." *In re Gurley*, 27 F.3d 551, 554, 31 USPQ2d 1130, 1132 (Fed. Cir. 1994).

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

9. **Claims 1-4, 6-12, 14 and 17-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over 'Reconfigurable, Retargetable Bignums' by White (White) in view of US 5,640,496 to Hardy et al. (Hardy).**

Regarding Claims 1 and 18: White discloses a computer-implemented method for processing numerical values in a computer program executable on a computer system, comprising: encapsulating in a large-integer datatype, large-integer data and associated operators (pg. 174, col. 1, par. 3 'indefinitely large integers—have been a part of Lisp for a long time'), wherein the large-integer data has runtime expandable precision (pg. 177, col. 2, par. 2 'allocated in units of at least one 32-bit word') and maximum precision is limited only by system memory availability (pg. 174, col. 1, par. 3 'indefinitely large integers); and overloading language-provided arithmetic, logical, and type conversion operators with the large-integer operators that operate on large-integer variables in combination with other datatypes, and programmed usage of a variable of the large-integer datatype is equivalent to and interoperable with a variable of a system-defined integral datatype (pg. 174, col. 1, par. 3 - col. 2, par. 1 'a smooth, user invisible transition between ... fixnums—and those of larger size');

White does not disclose establishing a plurality of available storage nodes available for allocation to large-integer data; or allocating a subset of the plurality of available storage nodes for a large-integer variable, the subset being an allocated plurality of storage nodes, or storing a numerical value in the allocated plurality of storage nodes or forming a linked list of the allocated plurality of storage nodes. He does however disclose that Bignums are allocated memory in pre-sized chunks (pg. 176, col. 2, par. 3 'primitives ... to allocate memory for one of a given size').

Hardy discloses establishing a plurality of available storage nodes available for allocation (col. 8, lines 18-21 'A free list of available nodes'); and allocating a subset of the plurality of available storage nodes for a variable (col. 8, lines 8-10 'memory will be allocated ... for the pixel value nodes'), the subset being an allocated plurality of storage nodes, and forming a linked list of the allocated plurality of storage nodes (col. 8, lines 8-10 'nodes of the linked list) in an analogous art for the purpose of memory management (Hardy col. 8, line 4 'memory must be managed efficiently').

It would have been obvious to a person of ordinary skill in the art at the time of the invention to use Hardy's methods of memory allocation/de-allocation (col. 8, lines 4-27) with White's invention (pg. 176, col. 2, par. 3) to provide memory space for White's Bignums (pg. 177, par. 1 'Bignums are allocated in units of at least one 32-bit word') because one of ordinary skill in the art would have been motivated to provide an efficient memory management system (Hardy col. 8, line 4 'memory must be managed efficiently') to support White's disclosure of memory allocation (pg. 176, col. 2, par. 3).

Regarding Claim 2: The rejection of claim 1 is incorporated; further White discloses that COMMON LISP is the base language for his invention (pg. 175, col. 1, par. 3 'other commercially available Common Lisp implementations').

"Common Lisp The Language, 2nd Edition" by Guy Steel et al. (Common Lisp) teaches converting a character string into large-integer data in response to a constant definition statement (sec. 5.1.1, par 1 'all numbers ... are self-evaluation forms. When such an object is evaluated, that object ... is returned as the value of the form').

Therefore, through his use of COMMON LISP, White inherently discloses converting a character string into large-integer data in response to a constant definition statement.

Regarding Claim 3: The rejection of claim 2 is incorporated; further White discloses that COMMON LISP is the base language for his invention (pg. 175, col. 1, par. 3 'other commercially available Common Lisp implementations').

Common Lisp teaches converting large-integer data to and from a character string for input (sec. 22.1.1 par. 16 'After the entire token is read in, it will be interpreted either as ... or number'), output (sec. 22.1.6, par. 2-3 'How an expression is printed depends on its data type'), and serialization (sec. 22.1.1 par. 16 'it begins an extended token. After the entire token is read in').

Therefore, through his use of COMMON LISP, White inherently discloses converting large-integer data to and from a character string for input, output, and serialization.

Regarding Claim 4: The rejection of claim 1 is incorporated; further, White discloses that COMMON LISP is the base language for his invention (pg. 175, col. 1, par. 3 'other commercially available Common Lisp implementations').

Common Lisp teaches converting input data from language-provided input functions to large-integer data (sec. 22.1.1 par. 15 'After the entire token is read in, it will be interpreted either as ... or number'); and converting large-integer data to a format compatible with language-provided output functions (sec. 22.1.6, par. 2-3 'How an expression is printed depends on its data type').

Therefore, through his use of COMMON LISP, White inherently discloses converting input data from language-provided input functions to large-integer data and converting large-integer data to a format compatible with language-provided output functions.

Regarding Claim 6: The rejection of claim 1 is incorporated; further, White discloses allocating a selected number of bits for each storage node in response to a program-specified parameter (pg. 178, col. 1, par. 3 'the size of a bigit, ... will vary from implementation to implementation').

Regarding Claim 7: The rejection of claim 1 is incorporated; further, White discloses dynamically allocating a number of storage nodes for storage of the numerical value as a function of a size of the numerical value (pg. 177, col. 2, par 2 'Bignums are allocated in units of at least one 32-bit word').

Regarding Claim 8: The rejection of claim 7 is incorporated; further, White discloses storing in each node that is allocated to a large-integer variable, a subset of bit values that represent a numerical value (pg. 178, col. 1, par. 3 'a 'bigit' is a 'bignum digit' and is thus an integer between 0 and R-1 for some positive radix R').

Regarding Claim 9: The rejection of claim 8 is incorporated; further White explicitly discloses allocating a storage node to a large-integer variable while performing a large-integer operation that generates a numerical value and stores the numerical value in the variable (pg. 180, col. 1 'allocating memory space for the result of a bignum-by-bignum multiplication'), if a number of bit values required to represent the numerical value exceeds storage available in storage nodes allocated to the large-integer variable pg. 180, col. 1 'We don't want the multiplication routine to ... be caught short by one bit'), and therefore inherently discloses maintaining a set of available storage nodes that are not allocated to any large-integer variable from which to allocate storage nodes. However White does not explicitly disclose returning to the set of available storage nodes a storage node allocated to a large-integer variable while performing a large-integer operation that generates a numerical value for storage in the variable, if a number of bit values required to represent the numerical value is less than storage available in storage nodes allocated to the variable, but does disclose that 'We don't want ... to allocate extra space needlessly' (pg. 180, col. 1) and "The Art of Computer Programming" by Knuth, from which White derives his division algorithm (pg. 180, col. 2 'The division algorithm is essentially 'Algorithm D"), teaches, 'division of an (m + n)-place integer by an n-place integer, giving an (m + 1)-place quotient and an n-place remainder' (pg. 250, section 4.3.1).

Accordingly, It would have been obvious to a person of ordinary skill in the art at the time of the invention to return any 'extra' nodes to the pool of available storage nodes, when it is found that the number of nodes required to represent the numerical value (pg. 250, section 4.3.1 'm + 1') is less that the number actually allocated (pg. 250, section 4.3.1 'm + n') because one of ordinary skill in the art would have been motivated to collect the empty nodes (pg. 180, col. 1 'We don't want ... to allocate extra space needlessly').

Regarding Claim 10: The rejection of claim 9 is incorporated; further White discloses that COMMON LISP is the base language for his invention (pg. 175, col. 1, par. 3 'other commercially available Common Lisp implementations').

Common Lisp inherently discloses overloading language-provided memory allocation and de-allocation operators with large-integer operators that allocate and de-allocate storage nodes. COMMON LISP seeks to make the use of Bignums and Fixnums seamless (sec 2.1.1 par. 'Common Lisp is designed to hide this distinction'), additionally functions such as *let* make no distinction between the two (sec. 7.5, par. 4 "all of the variables varj are bound to the corresponding values'), thus overloading the memory handling functions.

Therefore, through his use of COMMON LISP, White inherently discloses overloading language-provided memory allocation and de-allocation operators with large-integer operators that allocate and de-allocate storage nodes.

Regarding Claim 11: The rejection of claim 1 is incorporated; further White discloses the algorithm used for division is defined in "The Art of Computer Programming, Vol. II" by Knuth (Knuth) (pg. 177, col. 1, par. 2 'The particular algorithms used are ...described in section 4.3.1 of [Knuth 1981])

Knuth teaches identifying a set of most-significant bits of the dividend and a set of least-significant bits of the dividend (pg. 257, Algorithm D '$v=(v_1v_2...v_n)_b$'); recursively performing a large-integer divide operation using the set of most-significant bits as the input dividend (pg. 257, Algorithm D Step D2 'a division of $(u_ju_{j+1}...u_{j+n})_b$', and returning a quotient and a remainder (Algorithm D step D4 'replace $(u_ju_{j+1}...u_{j+n})_b$ by $(u_ju_{j+1}...u_{j+n})_b$ minus q times $(v_1v_2...v_n)_b$'); finding a lower-part dividend as a function of the remainder and the set of least-significant bits (Algorithm D step D7 'increase j by one'); recursively performing a large-integer divide operation using the lower-part dividend (Algorithm D steps D7 'go back to D3'); and concurrently solving for the quotient and the remainder (Algorithm D step D4 'minus q times $(v_1v_2...v_n)_b$').

Therefore, through the use of the algorithm taught by Knuth, White implicitly discloses the limitations recited.

Regarding Claim 12: The rejection of claim 11 is incorporated; further White discloses identifying an optimal set of most-significant bits of the dividend and a set of least-significant bits of the dividend as a function of a number of bits that represent the dividend and a number of bits that represent the divisor (pg. 178, col. 1, par. 3 'the size of a bigit ... will vary from ... algorithm to algorithm').

Regarding Claim 14: The rejection of claim 1 is incorporated; further White discloses emulating fixed-bit arithmetic on variables of the large-integer data type (pg. 174, col. 2, par. 4 'a set of new primitive arithmetic operations that focus on ... bignums').

Regarding Claims 17 and 20: The rejections of claims 1 and 18 are incorporated; further White does not explicitly disclose a large-rational datatype, but does disclose that COMMON LISP is the base language for his invention (pg. 175, col. 1, par. 3 'other commercially available Common Lisp implementations').

Common Lisp teaches a rational datatype (sec. 2.1.2, par. 1 'Integers and ratios collectively constitute the type rational') where a ratio is the mathematical ratio of two integers (sec. 2.1.2, par. 1), and integers encompass bignums (sec. 2.1.1, par. 2 'an integer that is not a fixnum is called a bignum'), thereby teaching the limitations recited in the instant claim as noted in the rejections of claims 1 and 18.

Therefore, through his use of COMMON LISP, White inherently discloses a large-rational datatype.

Claim 13 is rejected under 35 U.S.C. 103(a) as being unpatentable over 'Reconfigurable, Retargetable Bignums' by White (White) in view of US 5,640,496 to Hardy et al. (Hardy) further in view of "Fast Recursive Division" by Burnikel et al. (Burnikel).

Regarding Claim 13: The rejection of claim 12 is incorporated; further White does not disclose identifying an optimal set of most-significant bits of the dividend and a set of least-significant bits of the dividend as a function of one-half a difference between the number of bits that represent the dividend and the number of bits that represent the divisor. However, White does disclose the possibility of using various algorithms (pg. 177, col. 1, par. 2 'we have investigated some more complex algorithms').

Burnikel teaches a division algorithm identifying an optimal set of most-significant bits of the dividend and a set of least-significant bits of the dividend as a function one-half a difference between the number of bits that represent the dividend and the number of bits that represent the divisor (pg. 4, par. 3 'dividing a 2n-digit number by an n-digit number ... split A into four parts ... of length n/2 each') in an analogous art for the purpose of improving the processing speed of a divide operation (pg. 1, par. 1 'our algorithm ... yields a speedup of more than 20%')

It would have been obvious to a person of ordinary skill in the art at the time of the invention to implement the division operation disclosed in White using the algorithm taught in Burnikel (pg. 4, Algorithm 1), because one of ordinary skill in the art would have been motivated to improve performance for a system where most of the division would be done for larger numbers (White pg. 177, col. 1, par. 2 'algorithms that do show a significant improvement in the asymptotic behaviors' and Burnikel pg. 4, par. 2 'Under the assumption that n is even and large').

Claims 15-16 and 19 are rejected under 35 U.S.C. 103(a) as being unpatentable over 'Reconfigurable, Retargetable Bignums' by White (White) in view of US 5,640,496 to Hardy et al. (Hardy) further in view of US 5,619,711 to Anderson (Anderson).

Regarding Claim 15: The rejection of claim 1 is incorporated; further White does not disclose transferring data associated with temporary variables of the large-integer datatype by moving pointers to the data.

Anderson teaches transferring data associated with temporary variables of the large-integer datatype by moving pointers to the data, in an analogous art for the purpose of avoiding fragmentation when expanding the data structure of a large-integer (col. 8, lines 59-61 'to avoid fragmentation, a linked list type of allocation may be used for the array')

It would have been obvious to a person of ordinary skill in the art at the time of the invention to utilize the techniques taught in Anderson (col. 8, lines 59-61) when updating large-integer-data as disclosed in White (pg. 174, col. 1, par. 3 'indefinitely large integers) because one of ordinary skill in the art would have been motivated to avoid fragmentation as taught in Anderson (col. 8, lines 59-61).

Regarding Claims 16 and 19: The rejections of claims 1 and 18 are incorporated; further White does not explicitly disclose a large-floating-point datatype, but does disclose that COMMON LISP is the base language for his invention (pg. 175, col. 1, par. 3 'other commercially available Common Lisp implementations').

Anderson teaches a large-floating-point datatype (col. 4, lines 15-17 'implementing infinite precision binary arithmetic') in an analogous art for the purpose of preserving numerical precision (col. 3, lines 15-17 'for preserving numerical precision').

It would have been obvious to a person of ordinary skill in the art at the time of the invention to use the teachings of Anderson to incorporate a large-floating-point datatype, using the techniques taught in Anderson (col. 4, lines 15-17), in a method similar to that disclosed for Bignums in White (pg. 174, col. 1, par. 3 'indefinitely large integers—have been a part of Lisp for a long time'), because one of ordinary skill in the art would have been motivated to 'provide special operation for floating point math' (Anderson col. 2, lines 5-8), thereby teaching the limitations recited in the instant claim as noted in the rejections of claims 1 and 18.

Claim 21 is rejected under 35 U.S.C. 103(a) as being unpatentable over 'Reconfigurable, Retargetable Bignums' by White (White) in view of US 5,640,496 to Hardy et al. (Hardy) further in view of US 6,078,994 to Carey (Carey).

Regarding Claim 21: The rejection of claim 1 is incorporated; further Hardy does not explicitly teach maintaining a minimum or maximum number of available storage nodes, but does teach maintaining a list of available storage nodes (col. 8, lines 18-21 'A freelist of available nodes 39 is kept within each memory block').

Carey teaches determining a total number of available storage nodes available for allocation to large-integer data (col. 7, lines 40-42 'maintains a counter of the number of entries on the free list'); allocating memory for a first number of available storage nodes, responsive to the total number being less than first threshold value, and establishing the first number of available storage nodes (col. 7, lines 44-46 'If the minimum threshold is met ... begins a collecting operation'); and halting collection of available storage nodes, responsive to the total number being greater than a second threshold value (col. 8, lines

44-46 'If this number is above a preset maximum threshold then processing [suspends]').

It would have been obvious to a person of ordinary skill in the art at the time of the invention track the number of nodes in Hardy's 'free list' (col. 8, lines 18-21 'A free list of available nodes') thereby maintaining at least a minimum number of nodes as taught in Carey (col. 7, lines 44-46 'the minimum threshold') in order to avoid processing delays (Carey col. 7, lines 40-41 'minimize processing delays ... when the free list becomes empty'). Further in Hardy's system, where nodes are explicitly returned to the 'free list' (col. 8, lines 22-23 'As nodes are removed from a linked list, they are returned to the free list') and not collected as in Carey (col. 7, lines 44-46 'begins a collecting operation'), it would have also been obvious to remove nodes from the 'free list' when the count exceeded the maximum threshold taught in Carey (col. 8, lines 44-46 'If this number is above a preset maximum threshold) in order to maintain an optimal number of nodes in the free list (Carey col. 8, lines 49-51 'to optimize the cash memory').

Conclusion

10. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. US 2001/0047361 to Martin et al. discloses a method of maintaining a free list.

11. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP

§ 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jason Mitchell whose telephone number is (571) 272-3728. The examiner can normally be reached on Monday-Thursday and alternate Fridays 7:30-5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571) 272-3719. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

[illegible]

Jason Mitchell
6/1/05

[illegible]

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100